Guía de uso de MATLAB

Se necesitan unos pocos comandos básicos para empezar a utilizar MATLAB. Esta pequeña guía explica dichos comandos fundamentales. Habrá que **definir** vectores y matrices para poder **modificarlos** y **operar** con ellos. Se trata de comandos cortos de alto nivel, porque MATLAB trabaja constantemente con matrices. Creo que les gustarán las posibilidades que les ofrece este software para realizar operaciones de álgebra lineal mediante una serie de instrucciones cortas:

definir E	*definir u*	*modificar E*	*multiplicar Eu*
E = eye(3)	u =E(:,1)	E(3,1)=5	v =E*u
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix}$

La palabra *eye* designa a la matriz identidad. La submatriz u = E(:,1) toma la primera columna de la anterior. La instrucción E(3, 1) = 5 coloca un 5 en el elemento (3, 1). El comando E* u multiplica las matrices E y u. Todos estos comandos se repiten en la lista que aparece a continuación. Aquí se presenta un ejemplo de cómo invertir una matriz y resolver un sistema lineal:

definir A	*definir b*	*invertir A*	*Resolver Ax=b*
A = ones(3) + eye(3)	b = A(:,3)	C = inv(A)	x = A\\b o x = C*b
$\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} .75 & -.25 & -.25 \\ -.25 & .75 & -.25 \\ -.25 & -.25 & .75 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Se sumó una matriz formada por unos a eye(3), y b es su tercera columna. A continuación, inv(A) genera la matriz inversa (normalmente en decimales, ya que para las fracciones se usa *format rat*). El sistema $Ax = b$ se resuelve mediante x = inv(A) * b, el método lento. El comando de la barra inversa x = A\b realiza la eliminación gaussiana si A es cuadrada y nunca calcula la matriz inversa. Cuando la parte derecha de b sea igual a la tercera columna de A, la solución para x tiene que ser [0 0 1]'. (**El símbolo de la transpuesta ' convierte a *x* en un vector de columna**.) Entonces A*x elige la tercera columna de A, y tenemos que $Ax = b$.

A continuación aparece una serie de comentarios, precedidos por el símbolo %:

% Los símbolos *a* y *A* son **diferentes**: MATLAB distingue por defecto entre unos casos y otros.

% Escribir *help slash* para obtener una explicación del modo de utilizar el símbolo de la barra inversa. La palabra *help* (ayuda) puede ir seguida de un símbolo o del nombre de un comando o de un archivo (de extensión .m) de MATLAB.

Nota: El nombre del comando aparece con una mayúscula inicial en la explicación que da *help*, pero debe escribirse en minúsculas al utilizarlo. La barra inversa $A \backslash b$ actúa de forma distinta cuando A no es cuadrada.

% Para ver los números con 16 dígitos, escribir *format long* (formato largo). El formato normal, *format short* (formato corto), muestra 4 dígitos decimales.

% Si se pone un punto y coma tras un comando, el programa no mostrará su resultado. A = ones(3); no mostrará la matriz identidad de 3 x 3.

% Utilizar la flecha del desplazamiento hacia arriba del cursor para volver a comandos anteriores.

Cómo introducir un vector de filas o de columnas

u = [2 4 5] tiene una fila con tres elementos (matriz de 1 x 3).
v = [2; 4; 5] tiene tres filas separadas por puntos y comas (matriz de 3 x 1).
v = [2 4 5]' o $v = u'$ transpone u para generar la misma v.
w = 2:5 define el vector de filas w = [2 3 4 5] mediante valores que aumentan sucesivamente en una unidad.
u = 1:2:7 asigna valores que aumentan en dos unidades para obtener u = [1 3 5 7]

Cómo definir una matriz (introduciendo las filas una por una)

A = [1 2 3; 4 5 6] tiene dos filas (el punto y coma siempre separa unas filas de otras).

A = [12 3

4 5 6] también genera la matriz A, pero es más difícil de escribir.

B = [1 2 3; 4 5 6]' es la **transpuesta** de A. Así pues, A^T es A' en MATLAB.

Cómo generar matrices especiales

diag(v) genera una matriz diagonal con el vector v como diagonal.

toeplitz(v) define una matriz simétrica de **diagonal constante** con v como primera fila y primera columna.

toeplitz(w, v) define una matriz simétrica de diagonal constante con w como primera

columna y v como primera fila.

ones(n) genera una matriz de $n \times n$ con todos los valores iguales a uno.

zeros(n) genera una matriz de $n \times n$ con todos los valores iguales a cero.

eye(n) genera una matriz identidad de $n \times n$.

rand(n) genera una matriz de $n \times n$ con elementos de valor aleatorio entre 0 y 1

(distribución uniforme).

randn(n) genera una matriz de $n \times n$ cuyos elementos siguen una distribución normal

(media 0 y varianza 1).

ones(m, n), zeros(m, n), rand(m, n) generan matrices de $m \times n$.

ones(size(A)), zeros(size(A)), eye(size(A)) generan matrices de la misma forma que A.

Cómo cambiar elementos en una matriz A dada

$A(3, 2) = 7$ coloca un 7 en el elemento (3, 2).
$A(3,:) = v$ sustituye los valores de la tercera fila por los de v.
$A(:, 2) = w$ sustituye los valores de la segunda columna por los de w.
El símbolo de los dos puntos : significa *todo* (todas las columnas o todas las filas).
$A([2\ 3],:) = A([3\ 2],:)$ intercambia las filas 2 y 3 de A.

Cómo crear submatrices de una matriz *A* de $m \times n$

$A(i, j)$ muestra el elemento (i, j) de la matriz A (escalar = matriz de 1×1).
$A(i, :)$ muestra la fila i-ésima de A (como vector de fila).
$A(:, j)$ muestra la columna j-ésima de A (como vector de columna).
$A(2{:}\ 4,3{:}\ 7)$ muestra las filas de la 2 a la 4 y las columnas de la 3 a la 7 (en forma de matriz de 3×5).
$A([2\ 4],:)$ muestra las filas 2 y 4 y todas las columnas (en forma de matriz de $2 \times n$).
$A(:)$ muestra una sola columna larga formada a partir de las columnas de A (matriz de $mn \times 1$).
triu(A) coloca ceros en todos los elementos por debajo de la diagonal (triangular superior).
tril(A) coloca ceros en todos lo elementos por encima de la diagonal (triangular inferior).

Multiplicación e inversión de matrices

$A * B$ da la matriz resultante del producto AB (si dicha operación es posible).

$A. * B$ da el producto elemento por elemento (si size(A) = size(B), es decir, si tienen el mismo tamaño)
inv(A) da A^{-1} si A es cuadrada e invertible.
pinv(A) da la pseudoinversa de A.
A\B da inv(A) * B si existe inv(A): *la barra inversa* es la división por la izquierda.
$x = A$\b da la solución de $Ax = b$ si existe inv(A).
¡Véase *help slash* cuando A sea una matriz rectangular!

Números y matrices asociados a *A*

det(A) es el **determinante** (si A es una matriz cuadrada).

rank(A) es el **rango** (número de pivotes = dimensión del espacio de filas y del espacio de columnas).

size(A) es el par de números [m n].

trace(A) es la **traza** = suma de los elementos de la diagonal = suma de autovalores.

null(A) es una matriz cuyas columnas $n - r$ forman una base ortogonal para el espacio nulo de A.

orth(A) es una matriz cuyas columnas r forman una base ortogonal para el espacio de columnas de A.

Ejemplos

E = eye(4); E(2, 1) = -3 crea una matriz de eliminación elemental de 4×4.
$E*A$ resta 3 veces la fila 1 de la fila 2 de A.
$B = [A\ b]$ crea una matriz aumentada con b como columna adicional.
E = eye(3); $P = E$([2 1 3],:) genera una matriz de permutación.
Nótese que triu(A) + tril(A) - diag(diag(A)) es igual a A.

Archivos .m incluidos en el programa para realizar la factorización de matrices (¡importantísimos!)

[L, U, P] = lu(A) produce tres matrices donde $PA = LU$.

e = eig(A) es un vector en el que se encuentran los valores propios de A.

[S, E] = eig(A) produce una matriz diagonal de autovalores E y una matriz de autovectores S donde $AS = SE$. Si A no es diagonalizable (no tiene suficientes autovectores), S no es invertible.

[Q, R] = qr(A) produce una matriz ortogonal Q de $m \times m$ y una triangular R de $m \times n$, siendo $A = QR$.

Creación de archivos de extensión .m

Son archivos con la terminación .m que MATLAB utiliza para trabajar con funciones y scripts. Un script es una secuencia de comandos que se pueden ejecutar a menudo y que se pueden guardar en un archivo de extensión .m para no tener que escribirlos de nuevo. Las demostraciones de MATLAB son un ejemplo de estos scripts. Fijémonos en la que lleva por nombre *house* (casa). La mayoría de las funciones de MATLAB están en realidad en archivos .m, y se pueden visualizar escribiendo *type xxx*, donde *xxx* es el nombre de la función.

Para elaborar sus propios scripts o funciones, deberán generar un nuevo archivo de texto con el nombre que ustedes quieran, siempre y cuando termine en .m, para que MATLAB lo reconozca. Este tipo de archivos se pueden crear, editar y guardar con cualquier editor de textos, como *emacs, EZ,* o *vi*. Un archivo de script es simplemente una lista de comandos de MATLAB. Cuando se escribe el nombre del archivo en el prompt de MATLAB, su contenido se ejecuta. Para que un archivo .m sea una función, tiene que empezar por la

palabra *function* seguida de las variables de salida entre paréntesis, el nombre de la función y las variables de entrada.

Ejemplos

```
function [C]=mult(A)
r=rank(A);
C =A' * A;
```

Guardar los comandos que aparecen arriba en un archivo de texto llamado mult.m. Esta función tomará la matriz *A* y mostrará solamente la matriz resultado *C*. La variable *r* no se muestra porque no se introdujo como variable de salida. Al final de los comandos se ha puesto ";" para que no aparezcan en la ventana de MATLAB cada vez que se ejecutan. Esto resulta útil para trabajar con matrices grandes. Éste es otro ejemplo:

```
function [V, D, r]=properties(A)
% Esta función calcula el rango, autovalores y autovectores de A
[m, n]=size(A);
 if m==n
[V, D]=eig(A); r=rank(A); else
 disp('Error: La matriz debe ser cuadrada');
 end
```

Aquí, la función toma la matriz *A* como entrada y sólo muestra dos matrices y el rango como salida. El % se utiliza para marcar un comentario. La función comprueba si la matriz de entrada es cuadrada y luego calcula el rango, los autovalores y autovectores de la matriz *A*. Al escribir *properties(A)* sólo se obtendrá la primera salida, *V*, la matriz de autovectores. Es necesario escribir *[V,D,r]=properties(A)* para obtener las tres salidas.

Llevar un diario de trabajo

El comando *diary('file')* ordena a MATLAB que grabe todas la operaciones que se realizan en su ventana y que guarde los resultados en el archivo de texto de nombre *'file'*. Al escribir *diary on* y *diary off* activa y desactiva la grabación. Los archivos del diario se pueden visualizar mediante un editor de textos, o se pueden imprimir con *lpr* en unix. En MATLAB se pueden visualizar utilizando el comando type *file*.

Guardar variables y matrices

La instrucción *diary* graba tanto los comandos introducidos como la salida de MATLAB, pero no graba los valores de las variables y matrices. La orden *whos* elabora un lista de dichas variables, así como de las dimensiones de la matrices. El comando *save 'xxx'* guarda las matrices y variables de esta lista en un archivo denominado xxx. MATLAB etiqueta estos archivos con una extensión .mat, en lugar de con la .m que usa para scripts y

funciones. MATLAB podrá leer posteriormente los archivos xxx.mat mediante la orden *load xxx*.

Gráficos

El comando más simple es plot(*x*, *y*), que utiliza dos vectores, *x* e *y*, de la misma longitud. Éste dibujará los puntos (*xi*, *yi*) y los unirá mediante rectas continuas.

Si no se le da ningún vector *x*, MATLAB asume que *x*(*i*) = *i*. A continuación plot(*y*) recibe el mismo espacio en el eje de las *x*: los puntos son (*i*, *y*(*i*)).

Se pueden cambiar el tipo y color de la línea que une los puntos mediante un tercer argumento. Si este argumento no existe, MATLAB dibuja por defecto una línea continua de color negro "-". Introduciendo *help plot* se obtienen muchas opciones, aquí sólo indicamos unas pocas:

MATLAB 5: plot(*x*, *y*,'*r*+ :') dibuja *r* en rojo, los puntos en forma de + y unidos por línea de puntos.

MATLAB 4: plot(*x*, *y*,' --') dibuja una línea discontinua y plot(x, *y*,'·'), una línea de puntos.

Se pueden omitir las líneas y representar sólo los puntos discretos de distintas formas:

plot(*x*, *y*,' o') dibuja círculos. Otras opciones son '+', 'x' o '*'.
Para obtener dos gráficas en los mismos ejes, utilizar plot(*x*, *y*, *X*, *Y*). Sustituyendo *plot* por *loglog*, *semilogy* o *semilogx*, se cambian uno o ambos ejes a la escala logarítimica. El comando axis([*a b c d*]) ajusta el tamaño del gráfico al del rectángulo $a \le x \le b$, $c \le y \le d$. Para dar título al gráfico o marcar los ejes de las *x* o de las *y*, se escribe entre comillas la etiqueta deseada, como en los ejemplos siguientes:

title ('altura del satélite') *xlabel* ('tiempo en segundos') *ylabel* ('altura en metros')

El comando *hold* conserva el gráfico anterior mientras se dibuja uno nuevo. Al repetir *hold*, se borra la pantalla. Para imprimir o guardar la pantalla de gráficos en un archivo, véase *help print* o ejecútese print –Pnombre de la impresora print –d nombre del archivo.